Stats Bootcamp Day 8 - Gaussian Processes

Your Name

9/15/2021

You may see a message prompting you to install a package to run this code, click install when prompted for the code to run properly

Recap + Today's Material

Previously we've discussed regression in the context of independent errors. However, this independence is rarely the case in spatial or time series settings. For example, home prices in a certain neighborhood may all be higher than purely what the characteristics of the homes dictate.

In the case of spatial data, we will use Gaussian processes to account for spatial covariances. To do this, we need to review the multivariate normal (gaussian) distribution.

Multivariate Normal

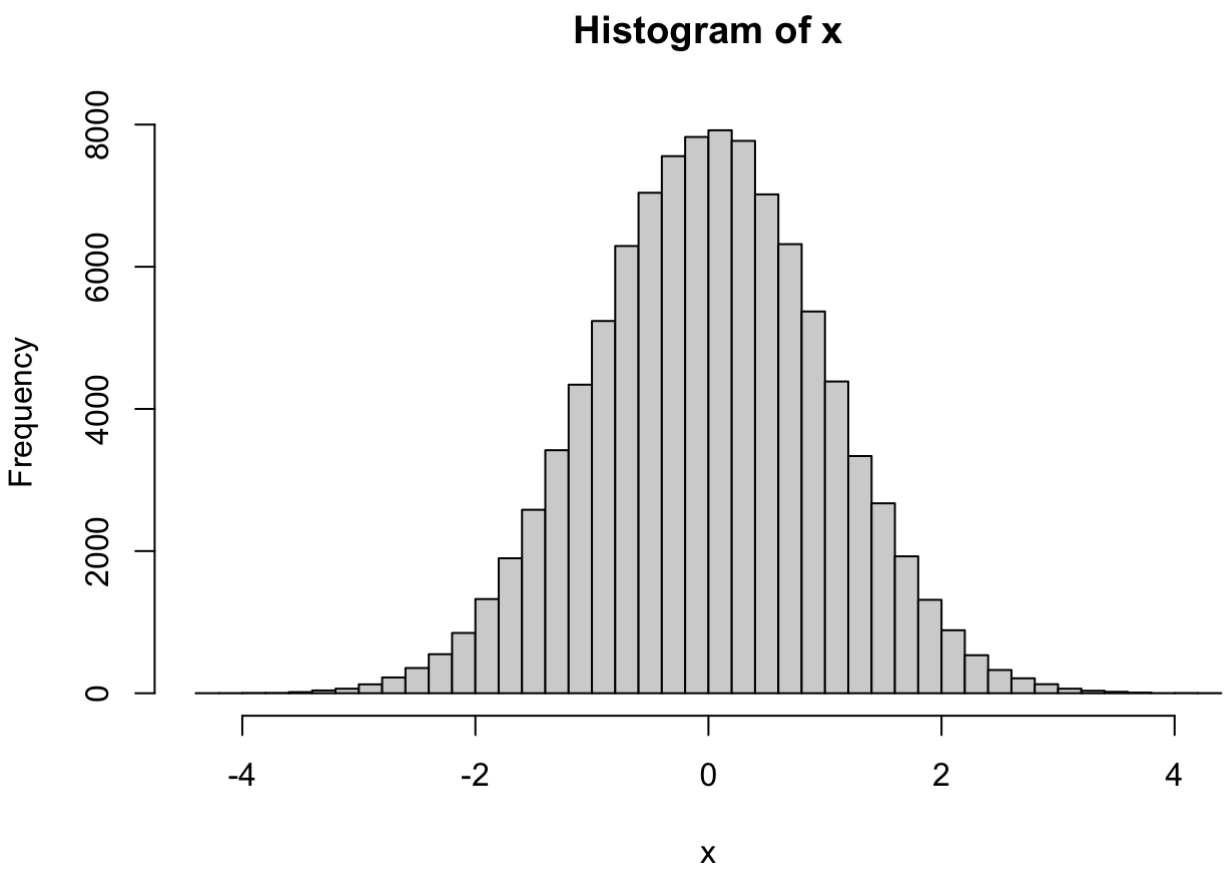
Let's stay in 2 dimensions initially to help visualize things

Let $x, y \sim N(0, 1)$

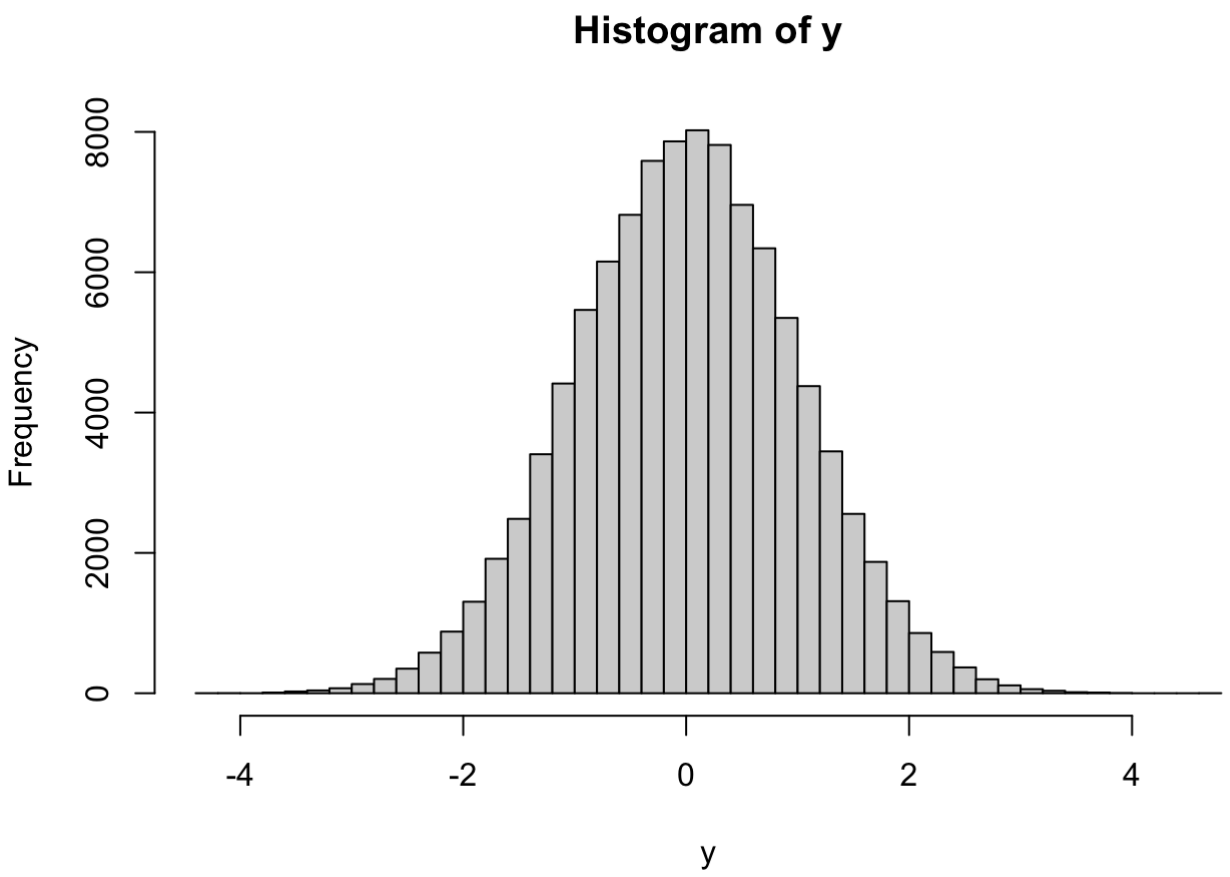
```
n=100000

#set x and y as seperate distributions, set mean and standard deviation to zero
x = rnorm(n, mean=0, sd=1)
y = rnorm(n, mean=0, sd=1)

hist(x, breaks = 50)
```




```
hist(y, breaks=50)
```



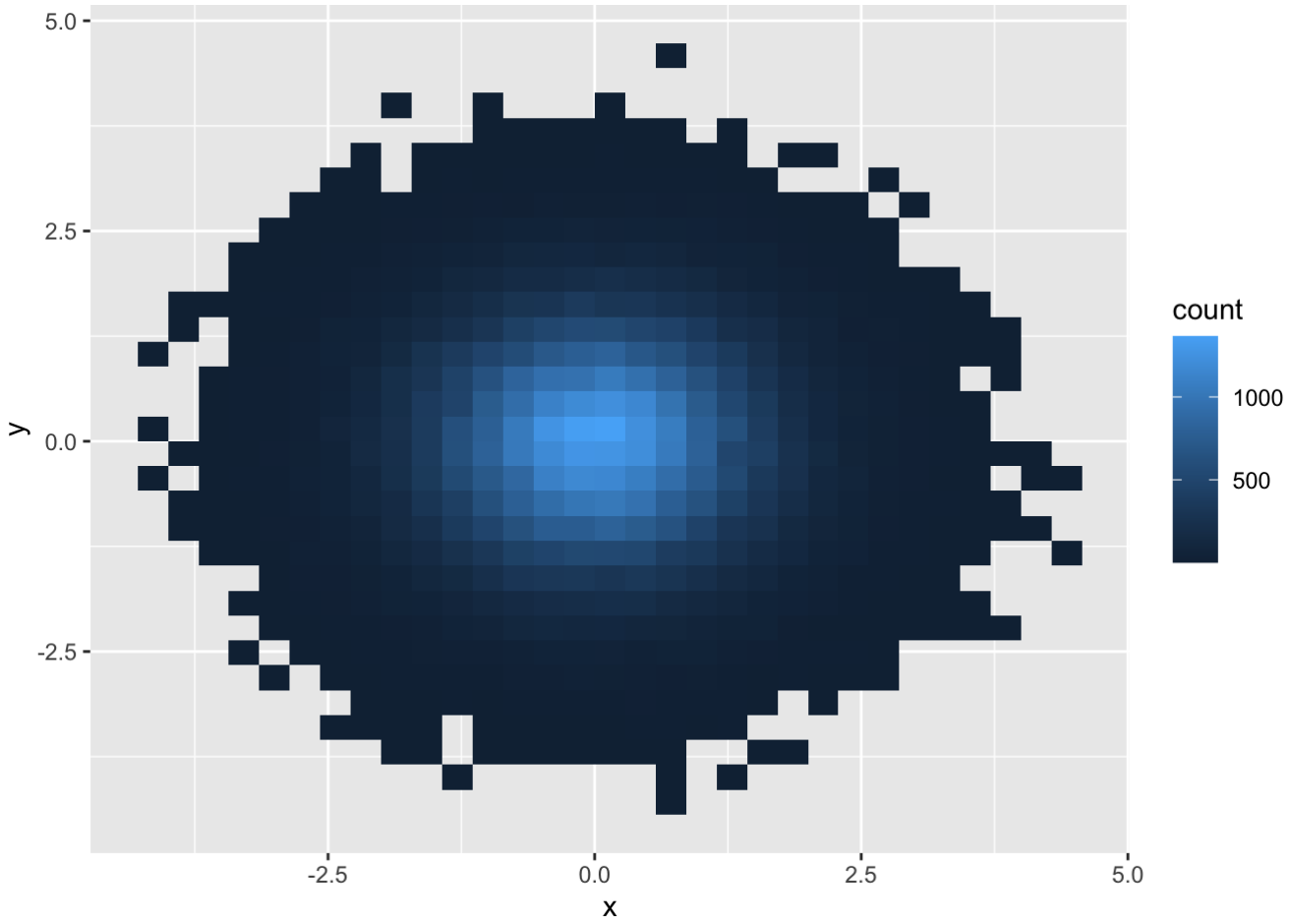
We can view the joint distribution of x,y in a 2D histogram, where color indicates frequency:

```
library(ggplot2) #load ggplot, which can be used to create 2d histogram plots
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
df = data.frame(x,y) #turn our data into a data frame

#plot the data frame
ggplot(df, aes(x,y)) + stat_bin2d()
```




```
cor(x,y)
```

```
## [1] -0.002660507
```

Note that we generated x,y above as independent random variables. We can use the multivariate normal distribution to make x,y individually distributed as standard normals, but that remain correlated.

To generate this multivariate normal distribution, we need the mvtnorm package. First, let's define a covariance matrix:

```
sigma = cbind(c(1,0.5), c(0.5,1))
sigma
```

```
##      [,1] [,2]
## [1,]  1.0  0.5
## [2,]  0.5  1.0
```

And now, generate the data

You do not need to see an output from running this command chunk below, we are just downloading a library

```
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 4.0.2
```

```
xy_correlated = rmvnorm(n=n, mean=c(0,0), sigma=sigma)
```

First, let's see if we have the desired correlation/covariance (the same here, since unit variance):

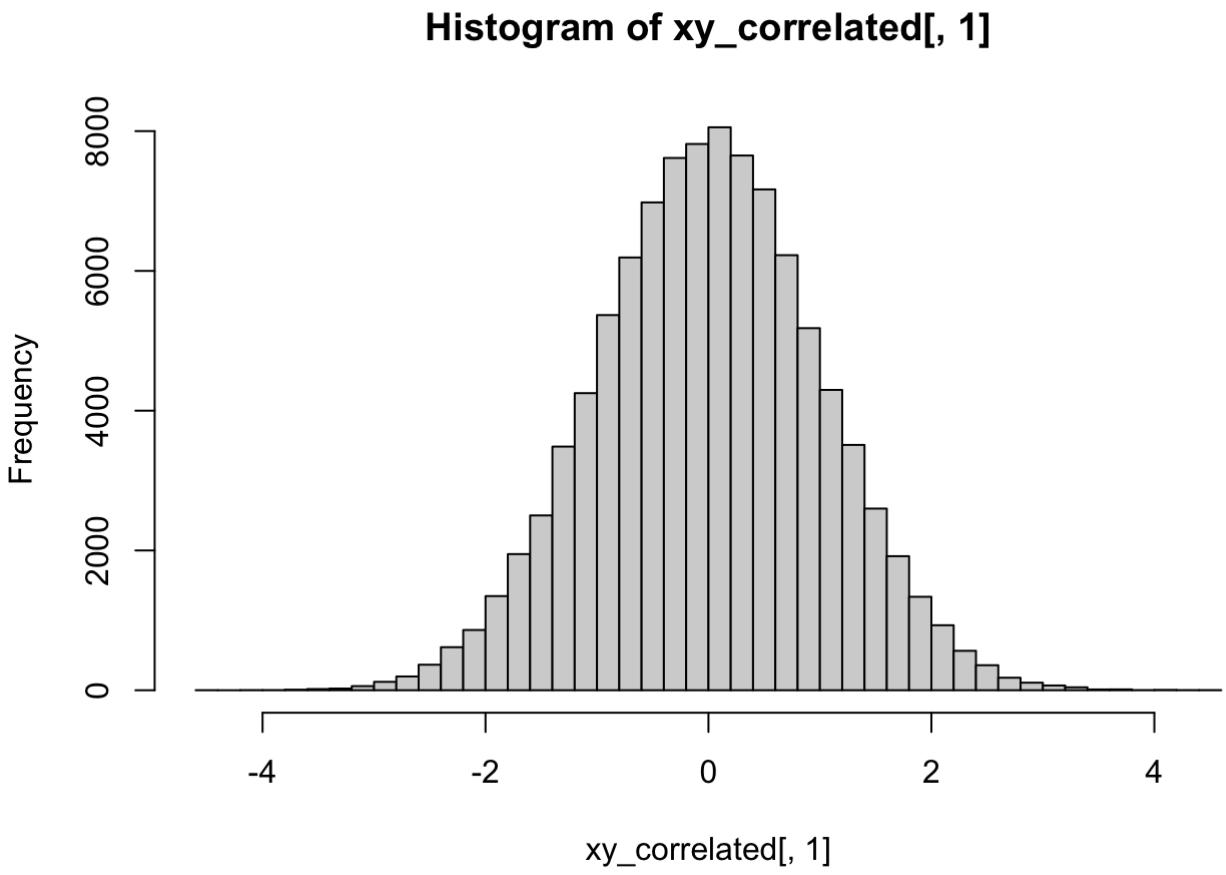
```
cor(xy_correlated[,1], xy_correlated[,2])
```

```
## [1] 0.5031768
```

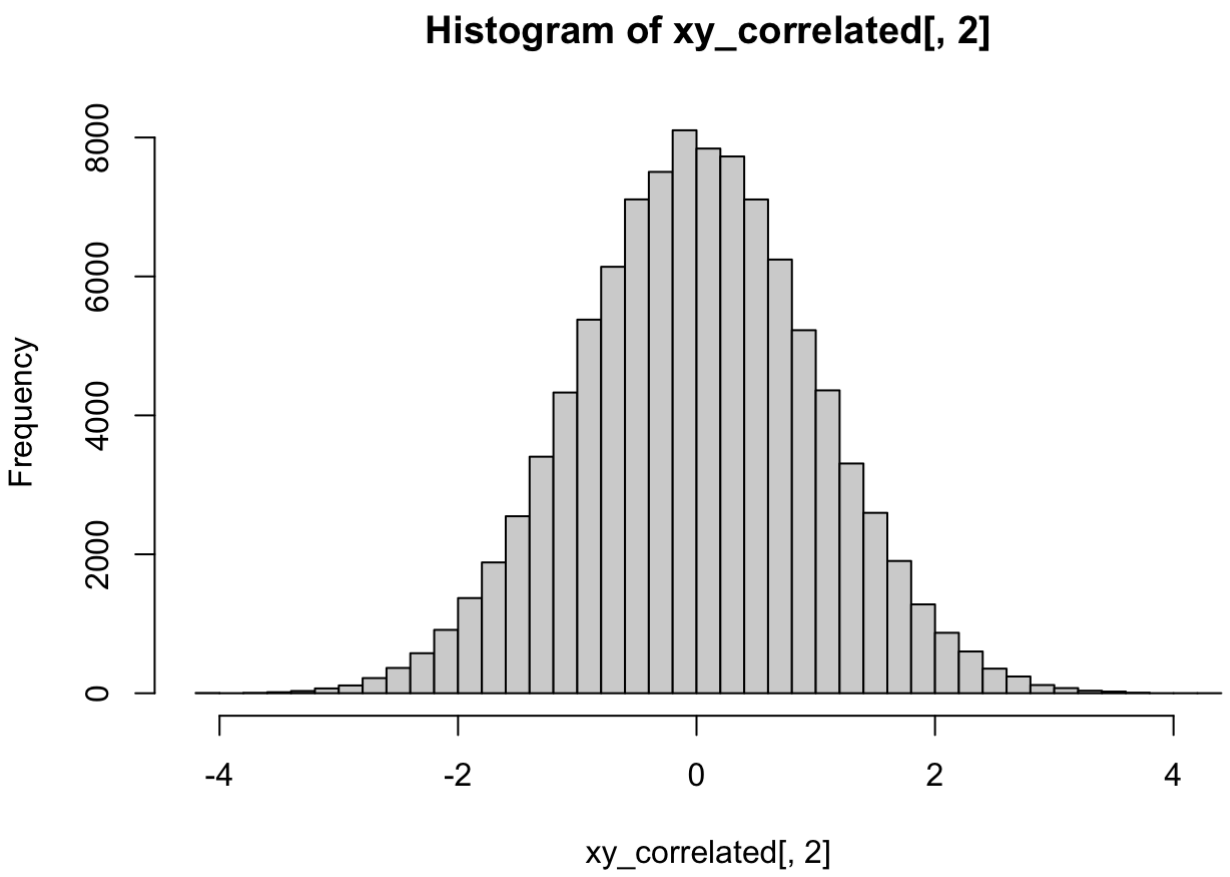
50%, like we hoped!

We can also check that each dimension still is N(0,1) when looked at individually:

```
hist(xy_correlated[,1], breaks=50)
```




```
hist(xy_correlated[,2], breaks=50)
```

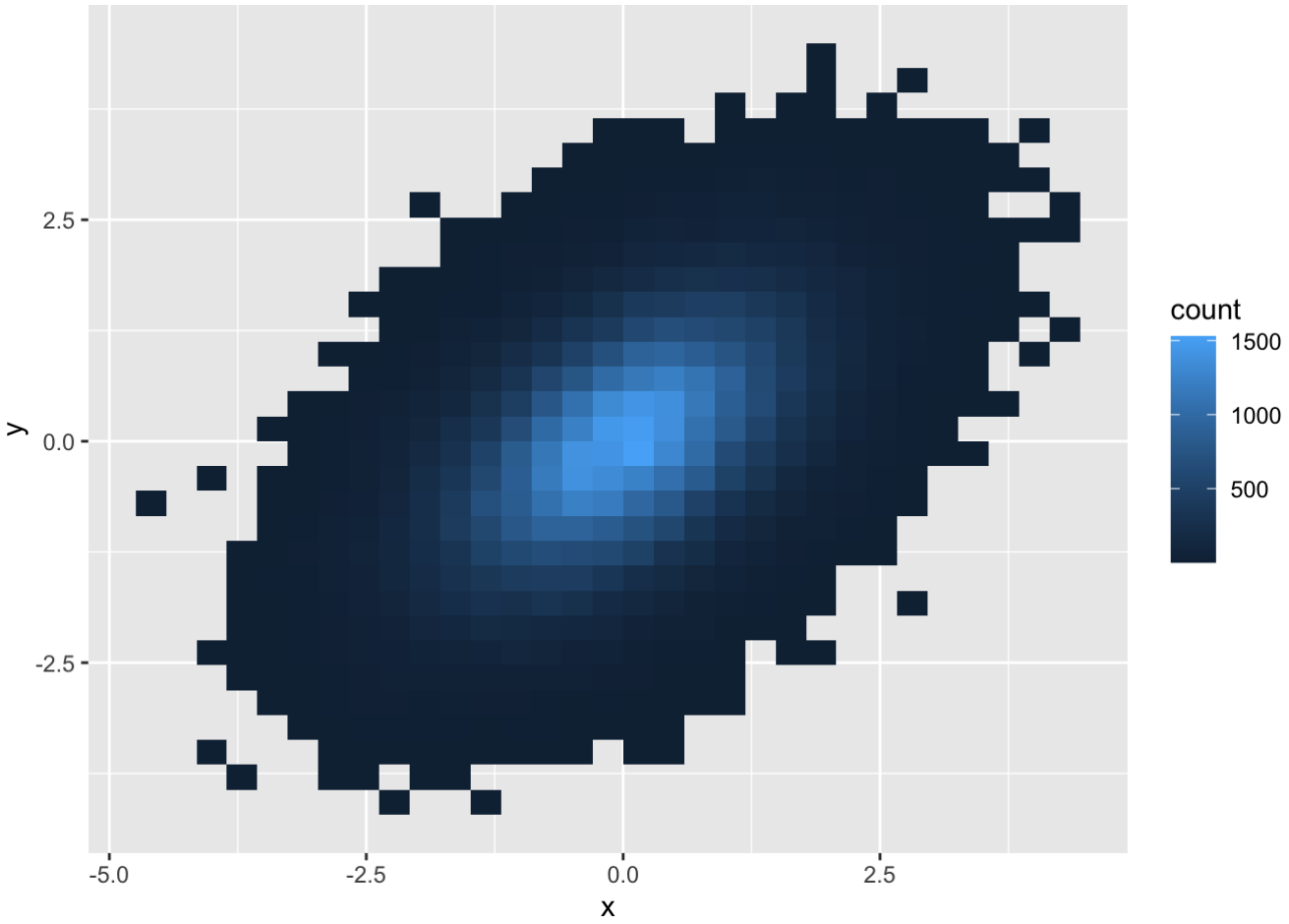


Definitely!

And now, let's see what the 2D-histogram looks like:

```
df2 = data.frame(xy_correlated)
colnames(df2) = c("x","y")

ggplot(df2, aes(x,y)) + stat_bin2d()
```



Woah! This looks very different. We have an ellipsoid shape here, since the y dimension's value is no longer independent of the x's. When x is high, then y will be similarly high

We can make the ellipsoid stretch in different directions by varying the covariance matrix. The main constraints on the covariance matrix is that it be positive semidefinite (PSD) and symmetric. We'll get a warning/error if we violate these conditions.

Check-in 1

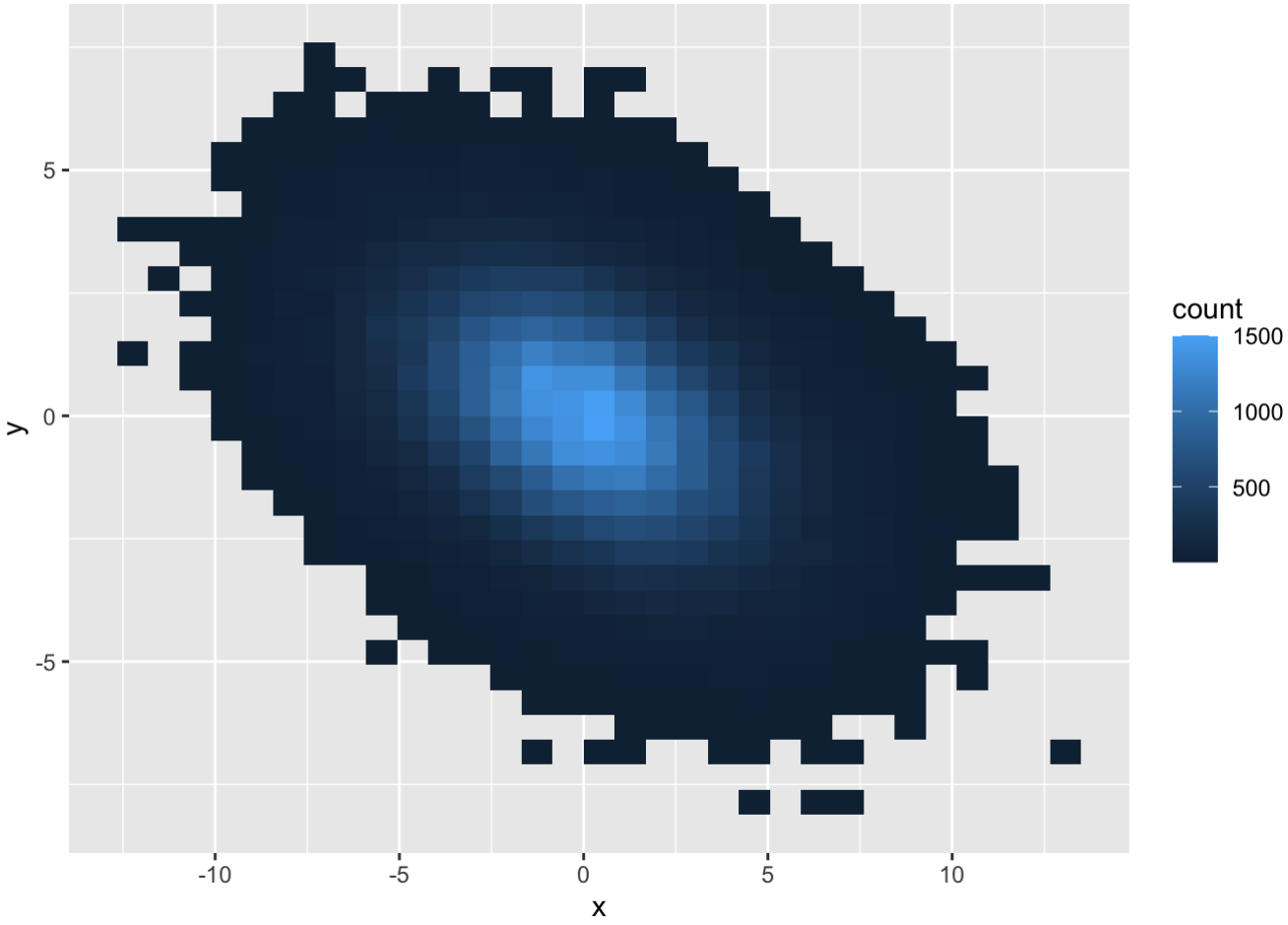
Use the above code to produce a similar 2D histogram plot, except with the ellipsoid stretching from the second quadrant to the fourth. Also try making the ellipsoid "bigger" (i.e. stretching out the x or y dimension)

```
#Add your answer here!
```

Check-in 1 Solution

```
sigma2 = cbind(c(8,-2), c(-2,3))
df3 = data.frame(rmvnorm(n=n, mean=c(0,0), sigma=sigma2))
colnames(df3) = c("x","y")

ggplot(df3, aes(x,y)) + stat_bin2d()
```



While we did all this in 2 dimensions, these notions generalize into n dimensions quite easily (just harder to graph).

Outline for Gaussian process

Suppose we are trying to model a temperatures in Maine across a range of locations. We can picture the model for the i'th location as follows:

$$T_i = f_i + \epsilon_i$$
$$cov(T_i, T_j) = cov(f_i, f_j) = g(distance(i,j))$$
$$\epsilon_i \sim_{iid} N(0, \sigma^2)$$

In other words, the temperature at location i can be composed into a spatial component f_i and random noise ϵ_i. The covariance between two locations is simply a function of the distance between those two locations.

We can make this framework simpler to write, using multivariate normal distributions:

$$T = f + \epsilon$$
$$\epsilon \sim N(0, \sigma^2 I)$$
$$f \sim N(\mu, \Sigma)$$
$$\Sigma_{i,j} = g(distance(i, j))$$

Since we usually assume ϵ is independent of f, we can rewrite T using the formula for the sum of two independent standard normals:

$$T \sim N(\mu, \Sigma + \sigma^2 I)$$

If we find a good function for $g(\cdot)$, then we will be able to both predict a new location as well as create a good statistical model to quantify uncertainty.

A natural choice for this g is a kernel function, which will help ensure that our ensuing covariance matrix is PSD and symmetric.

Let's constrain ourselves to the following kernel:

$$g(d) = c \cdot e^{-(d/l)^2}$$

l,c are parameters that we will have to choose / estimate

Let's load in the Maine dataset

Make sure that before you run this command you have downloaded the Maine_Temp dataset that we have provided you

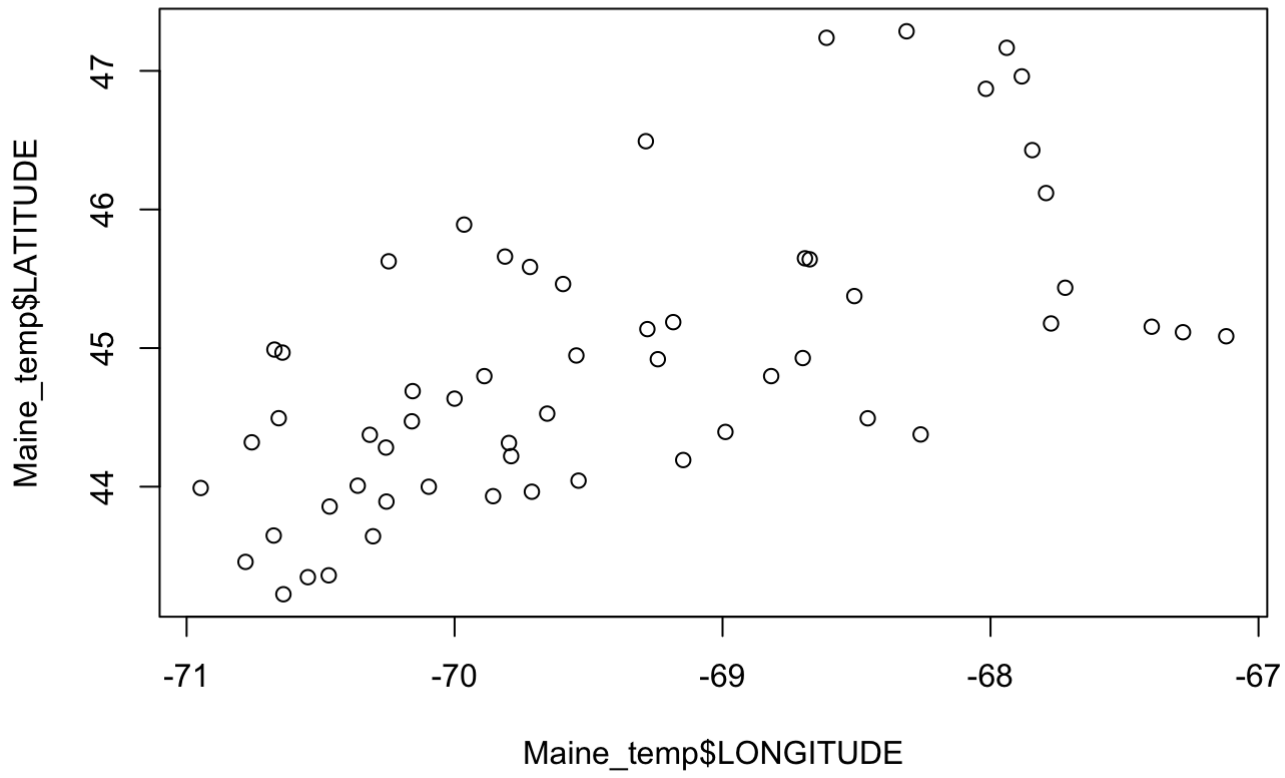
```
Maine_temp = read.csv("Maine_Temp.csv")
Maine_temp = Maine_temp[!is.na(Maine_temp$TMAX),]
Maine_temp
```

```
##         STATION                                        NAME LATITUDE
## 1   USC00170814                    BRASSUA DAM, ME US 45.66020
## 2   USC00171628                       CORINNA, ME US 44.91970
## 3   USC00179593              WEST ROCKPORT 1 NNW, ME US 44.19236
## 6   USW00014606     BANGOR INTERNATIONAL AIRPORT, ME US 44.79780
## 8   USW00014607   CARIBOU WEATHER FORECAST OFFICE, ME US 46.87050
## 10  USW00014609                HOULTON AIRPORT, ME US 46.11850
## 12  USW00014610    MILLINOCKET MUNICIPAL AIRPORT, ME US 45.64770
## 13  USC00175305           MILLINOCKET WASTEWATER, ME US 45.64040
## 14  USC00178817                       TURNER, ME US 44.28220
## 22  USC00170409                         BATH, ME US 43.93160
## 27  USC00170480                      BELFAST, ME US 44.39506
## 29  USC00177039                  RANGELEY 2 NW, ME US 44.98923
## 31  USC00171975              DOVER FOXCROFT WWTP, ME US 45.18720
## 34  USW00094623                WISCASSET AIRPORT, ME US 43.96361
## 36  USC00177037                       RANGELEY, ME US 44.96751
## 40  USC00174683    LINCOLN SANITARY DISTRICT WTP, ME US 45.37523
## 41  USW00094626 GREENVILLE MAINE FORESTRY SERVICE, ME US 45.46222
## 51  USC00173862                       HOLLIS, ME US 43.64759
## 53  USC00176856                       POLAND, ME US 44.00706
## 58  USR0000MMCF             MCFARLAND HILL MAINE, ME US 44.37690
## 59  USW00054772   FRYEBURG EASTERN SLOPES REGL AIRPORT, ME US 43.99056
## 60  USC00179720                    WINDHAM 2 NW, ME US 43.85670
## 65  USC00176722              PITTSTON FARMS NEPP, ME US 45.89088
## 67  USC00172765                    FARMINGTON, ME US 44.68938
## 68  USW00014605           AUGUSTA STATE AIRPORT, ME US 44.31550
## 70  USW00094644                    OLD TOWN 2 W, ME US 44.92810
## 71  USW00094645                LIMESTONE 4 NNW, ME US 46.96010
## 72  USC00173295                         GRAY, ME US 43.89250
## 73  USC00173570                      HARTFORD, ME US 44.37440
## 74  USC00170583                    BETHEL 6 SSE, ME US 44.32000
## 75  USC00171430                    CHURCHILL DAM, ME US 46.49250
## 77  USC00179151      WATERVILLE TREATMENT PLANT, ME US 44.52720
## 78  USC00172878                      FORT KENT, ME US 47.23860
## 83  USC00173046                      GARDINER, ME US 44.22020
## 84  USC00173567                       HARMONY, ME US 44.94660
## 87  USR0000MRCA             RACHEL CARSON MAINE, ME US 43.34720
## 88  USC00175460                      MOOSEHEAD, ME US 45.58530
## 90  USW00004836 FRENCHVILLE NORTHERN AROOSTOOK AIRPORT, ME US 47.28556
## 96  USC00174927                       MADISON, ME US 44.79760
## 98  USC00171131                    CAPE NEDDICK, ME US 43.22411
## 99  USC00177479                  SANFORD 2 NNW, ME US 43.45739
## 100 USC00177238                     ROBBINSTON, ME US 45.08533
## 101 USC00178965                    VAN BUREN 2, ME US 47.16640
## 102 USC00175736                      NEW SHARON, ME US 44.63520
## 104 USC00174086                         JACKMAN, ME US 45.62600
## 105 USC00179891                      WOODLAND, ME US 45.15420
## 108 USC00170833                    BRIDGEWATER, ME US 46.42820
## 116 USC00174193                   KENNEBUNKPORT, ME US 43.36050
## 118 USC00173261              GRAND LAKE STREAM, ME US 45.17760
## 119 USC00178792                    TOPSFIELD 2, ME US 45.43530
## 121 USR0000MMOO                  MOOSEHORN MAINE, ME US 45.11440
```

```
## 130 USC00175675                          NEWCASTLE, ME US 44.04360
## 131 USW00014764                    PORTLAND JETPORT, ME US 43.64222
## 132 USC00172443                          EAST SURRY, ME US 44.49340
## 133 USC00177336                       RUMFORD 6 SW, ME US 44.49404
## 134 USC00174745                 LIVERMORE FALLS 1 E, ME US 44.47160
## 136 USC00172048                              DURHAM, ME US 43.99960
## 137 USC00172440                 EAST SANGERVILLE, ME US 45.13690
##     LONGITUDE ELEVATION     DATE TMAX TMIN
## 1   -69.81190     325.8 1/1/2020   27   21
## 2   -69.24160      90.5 1/1/2020   29   23
## 3   -69.14678     110.0 1/1/2020   33   26
## 6   -68.81850      45.1 1/1/2020   36   24
## 8   -68.01730     190.2 1/1/2020   30   22
## 10  -67.79280     145.1 1/1/2020   30   23
## 12  -68.69250     123.7 1/1/2020   32   19
## 13  -68.67470     109.7 1/1/2020   25   19
## 14  -70.25610     172.2 1/1/2020   29   25
## 22  -69.85640       7.6 1/1/2020   31   28
## 27  -68.98925       6.1 1/1/2020   33   26
## 29  -70.67223     542.5 1/1/2020   27   24
## 31  -69.18410     112.8 1/1/2020   27   20
## 34  -69.71167      20.7 1/1/2020   39   28
## 36  -70.64220     466.3 1/1/2020   29   23
## 40  -68.50823      57.9 1/1/2020   28   22
## 41  -69.59528     316.1 1/1/2020   30   20
## 51  -70.67498      92.7 1/1/2020   32   22
## 53  -70.36149     173.7 1/1/2020   29   25
## 58  -68.26080     129.2 1/1/2020   33   26
## 59  -70.94750     135.6 1/1/2020   37   24
## 60  -70.46630     135.0 1/1/2020   30   23
## 65  -69.96446     328.0 1/1/2020   27   19
## 67  -70.15646     123.4 1/1/2020   34   23
## 68  -69.79720     107.0 1/1/2020   38   26
## 70  -68.70060      38.7 1/1/2020   35   25
## 71  -67.88330     224.6 1/1/2020   28   22
## 72  -70.25440     115.2 1/1/2020   35   26
## 73  -70.31630     214.9 1/1/2020   30   24
## 74  -70.75680     240.8 1/1/2020   32   24
## 75  -69.28630     284.7 1/1/2020   25   18
## 77  -69.65440      22.3 1/1/2020   29   26
## 78  -68.61195     185.9 1/1/2020   24   21
## 83  -69.78900      40.5 1/1/2020   32   28
## 84  -69.54560      97.5 1/1/2020   29   22
## 87  -70.54750       6.1 1/1/2020   43   31
## 88  -69.71870     316.1 1/1/2020   26   22
## 90  -68.31333     301.1 1/1/2020   27   22
## 96  -69.88900      67.1 1/1/2020   28   24
## 98  -70.63882      46.3 1/1/2020   38   30
## 99  -70.78022      88.1 1/1/2020   32   22
## 100 -67.12050      35.1 1/1/2020   33   27
## 101 -67.93960     139.0 1/1/2020   26   23
## 102 -70.00020     146.3 1/1/2020   29   25
```

```
## 104 -70.24600     370.0 1/1/2020   29   12
## 105 -67.39850      37.5 1/1/2020   32   29
## 108 -67.84420     128.0 1/1/2020   30   21
## 116 -70.46970       6.1 1/1/2020   38   30
## 118 -67.77420      88.4 1/1/2020   28   20
## 119 -67.72130     189.0 1/1/2020   26   23
## 121 -67.28190      64.0 1/1/2020   35   25
## 130 -69.53753      64.9 1/1/2020   33   27
## 131 -70.30444      13.7 1/1/2020   40   32
## 132 -68.45820      32.0 1/1/2020   36   26
## 133 -70.65627     192.0 1/1/2020   30   24
## 134 -70.15940     115.8 1/1/2020   30   21
## 136 -70.09630      42.7 1/1/2020   31   24
## 137 -69.28060     193.5 1/1/2020   34   21
```

```
plot(Maine_temp$LONGITUDE, Maine_temp$LATITUDE)
```




```
Maine_temp
```

```
##         STATION                                        NAME LATITUDE
## 1   USC00170814                    BRASSUA DAM, ME US 45.66020
## 2   USC00171628                       CORINNA, ME US 44.91970
## 3   USC00179593              WEST ROCKPORT 1 NNW, ME US 44.19236
## 6   USW00014606     BANGOR INTERNATIONAL AIRPORT, ME US 44.79780
## 8   USW00014607   CARIBOU WEATHER FORECAST OFFICE, ME US 46.87050
## 10  USW00014609                HOULTON AIRPORT, ME US 46.11850
## 12  USW00014610    MILLINOCKET MUNICIPAL AIRPORT, ME US 45.64770
## 13  USC00175305           MILLINOCKET WASTEWATER, ME US 45.64040
## 14  USC00178817                       TURNER, ME US 44.28220
## 22  USC00170409                         BATH, ME US 43.93160
## 27  USC00170480                      BELFAST, ME US 44.39506
## 29  USC00177039                  RANGELEY 2 NW, ME US 44.98923
## 31  USC00171975              DOVER FOXCROFT WWTP, ME US 45.18720
## 34  USW00094623                WISCASSET AIRPORT, ME US 43.96361
## 36  USC00177037                       RANGELEY, ME US 44.96751
## 40  USC00174683    LINCOLN SANITARY DISTRICT WTP, ME US 45.37523
## 41  USW00094626 GREENVILLE MAINE FORESTRY SERVICE, ME US 45.46222
## 51  USC00173862                       HOLLIS, ME US 43.64759
## 53  USC00176856                       POLAND, ME US 44.00706
## 58  USR0000MMCF             MCFARLAND HILL MAINE, ME US 44.37690
## 59  USW00054772   FRYEBURG EASTERN SLOPES REGL AIRPORT, ME US 43.99056
## 60  USC00179720                    WINDHAM 2 NW, ME US 43.85670
## 65  USC00176722              PITTSTON FARMS NEPP, ME US 45.89088
## 67  USC00172765                    FARMINGTON, ME US 44.68938
## 68  USW00014605           AUGUSTA STATE AIRPORT, ME US 44.31550
## 70  USW00094644                    OLD TOWN 2 W, ME US 44.92810
## 71  USW00094645                LIMESTONE 4 NNW, ME US 46.96010
## 72  USC00173295                         GRAY, ME US 43.89250
## 73  USC00173570                      HARTFORD, ME US 44.37440
## 74  USC00170583                    BETHEL 6 SSE, ME US 44.32000
## 75  USC00171430                   CHURCHILL DAM, ME US 46.49250
## 77  USC00179151      WATERVILLE TREATMENT PLANT, ME US 44.52720
## 78  USC00172878                      FORT KENT, ME US 47.23860
## 83  USC00173046                      GARDINER, ME US 44.22020
## 84  USC00173567                       HARMONY, ME US 44.94660
## 87  USR0000MRCA             RACHEL CARSON MAINE, ME US 43.34720
## 88  USC00175460                      MOOSEHEAD, ME US 45.58530
## 90  USW00004836 FRENCHVILLE NORTHERN AROOSTOOK AIRPORT, ME US 47.28556
## 96  USC00174927                       MADISON, ME US 44.79760
## 98  USC00171131                   CAPE NEDDICK, ME US 43.22411
## 99  USC00177479                  SANFORD 2 NNW, ME US 43.45739
## 100 USC00177238                     ROBBINSTON, ME US 45.08533
## 101 USC00178965                    VAN BUREN 2, ME US 47.16640
## 102 USC00175736                     NEW SHARON, ME US 44.63520
## 104 USC00174086                        JACKMAN, ME US 45.62600
## 105 USC00179891                     WOODLAND, ME US 45.15420
## 108 USC00170833                    BRIDGEWATER, ME US 46.42820
## 116 USC00174193                   KENNEBUNKPORT, ME US 43.36050
## 118 USC00173261              GRAND LAKE STREAM, ME US 45.17760
## 119 USC00178792                    TOPSFIELD 2, ME US 45.43530
## 121 USR0000MMOO                  MOOSEHORN MAINE, ME US 45.11440
```

```
## 130 USC00175675                          NEWCASTLE, ME US 44.04360
## 131 USW00014764                    PORTLAND JETPORT, ME US 43.64222
## 132 USC00172443                          EAST SURRY, ME US 44.49340
## 133 USC00177336                       RUMFORD 6 SW, ME US 44.49404
## 134 USC00174745                 LIVERMORE FALLS 1 E, ME US 44.47160
## 136 USC00172048                              DURHAM, ME US 43.99960
## 137 USC00172440                 EAST SANGERVILLE, ME US 45.13690
##     LONGITUDE ELEVATION     DATE TMAX TMIN
## 1   -69.81190     325.8 1/1/2020   27   21
## 2   -69.24160      90.5 1/1/2020   29   23
## 3   -69.14678     110.0 1/1/2020   33   26
## 6   -68.81850      45.1 1/1/2020   36   24
## 8   -68.01730     190.2 1/1/2020   30   22
## 10  -67.79280     145.1 1/1/2020   30   23
## 12  -68.69250     123.7 1/1/2020   32   19
## 13  -68.67470     109.7 1/1/2020   25   19
## 14  -70.25610     172.2 1/1/2020   29   25
## 22  -69.85640       7.6 1/1/2020   31   28
## 27  -68.98925       6.1 1/1/2020   33   26
## 29  -70.67223     542.5 1/1/2020   27   24
## 31  -69.18410     112.8 1/1/2020   27   20
## 34  -69.71167      20.7 1/1/2020   39   28
## 36  -70.64220     466.3 1/1/2020   29   23
## 40  -68.50823      57.9 1/1/2020   28   22
## 41  -69.59528     316.1 1/1/2020   30   20
## 51  -70.67498      92.7 1/1/2020   32   22
## 53  -70.36149     173.7 1/1/2020   29   25
## 58  -68.26080     129.2 1/1/2020   33   26
## 59  -70.94750     135.6 1/1/2020   37   24
## 60  -70.46630     135.0 1/1/2020   30   23
## 65  -69.96446     328.0 1/1/2020   27   19
## 67  -70.15646     123.4 1/1/2020   34   23
## 68  -69.79720     107.0 1/1/2020   38   26
## 70  -68.70060      38.7 1/1/2020   35   25
## 71  -67.88330     224.6 1/1/2020   28   22
## 72  -70.25440     115.2 1/1/2020   35   26
## 73  -70.31630     214.9 1/1/2020   30   24
## 74  -70.75680     240.8 1/1/2020   32   24
## 75  -69.28630     284.7 1/1/2020   25   18
## 77  -69.65440      22.3 1/1/2020   29   26
## 78  -68.61195     185.9 1/1/2020   24   21
## 83  -69.78900      40.5 1/1/2020   32   28
## 84  -69.54560      97.5 1/1/2020   29   22
## 87  -70.54750       6.1 1/1/2020   43   31
## 88  -69.71870     316.1 1/1/2020   26   22
## 90  -68.31333     301.1 1/1/2020   27   22
## 96  -69.88900      67.1 1/1/2020   28   24
## 98  -70.63882      46.3 1/1/2020   38   30
## 99  -70.78022      88.1 1/1/2020   32   22
## 100 -67.12050      35.1 1/1/2020   33   27
## 101 -67.93960     139.0 1/1/2020   26   23
## 102 -70.00020     146.3 1/1/2020   29   25
```

```
## 104 -70.24600     370.0 1/1/2020   29   12
## 105 -67.39850      37.5 1/1/2020   32   29
## 108 -67.84420     128.0 1/1/2020   30   21
## 116 -70.46970       6.1 1/1/2020   38   30
## 118 -67.77420      88.4 1/1/2020   28   20
## 119 -67.72130     189.0 1/1/2020   26   23
## 121 -67.28190      64.0 1/1/2020   35   25
## 130 -69.53753      64.9 1/1/2020   33   27
## 131 -70.30444      13.7 1/1/2020   40   32
## 132 -68.45820      32.0 1/1/2020   36   26
## 133 -70.65627     192.0 1/1/2020   30   24
## 134 -70.15940     115.8 1/1/2020   30   21
## 136 -70.09630      42.7 1/1/2020   31   24
## 137 -69.28060     193.5 1/1/2020   34   21
```

Calculate pairwise geographic distances between locations:

```
library(geosphere)
```

```
## Warning: package 'geosphere' was built under R version 4.0.2
```

```
geo.dist = distm(Maine_temp[,c(4,3)]) / 100000 # dividing by large num helps numeric sta
bility
geo.dist[1:5,1:5]
```

```
##           [,1]      [,2]      [,3]      [,4]     [,5]
## [1,] 0.0000000 0.9367121 1.7136308 1.2357858 1.929658
## [2,] 0.9367121 0.0000000 0.8117462 0.3608146 2.367271
## [3,] 1.7136308 0.8117462 0.0000000 0.7216645 3.104455
## [4,] 1.2357858 0.3608146 0.7216645 0.0000000 2.386346
## [5,] 1.9296580 2.3672707 3.1044552 2.3863464 0.000000
```

Let's write a function that calculates the covariance matrix, for a given c,l:

Do not expect an output out of the code chunk below as we are just creating the function, not executing it yet

```
calc_cov_matrix = function(a) {
  size = nrow(geo.dist)
  cov_matrix = matrix(nrow=size, ncol=size)

  for (i in 1:size) {
    for (j in 1:size) {

      d = geo.dist[i,j]

      cov_matrix[i,j] = a[1] * exp(-(d/a[2])^2)
    }
  }
  return(cov_matrix)
}
```

Here we are executing the function we created above, do not expect to see a result

```
calc_cov_matrix(c(1,2))[1:5,1:5]
```

```
##           [,1]      [,2]       [,3]      [,4]       [,5]
## [1,] 1.0000000 0.8030347 0.47992156 0.6826373 0.39420075
## [2,] 0.8030347 1.0000000 0.84812015 0.9679772 0.24635230
## [3,] 0.4799216 0.8481202 1.00000000 0.8779199 0.08986825
## [4,] 0.6826373 0.9679772 0.87791990 1.0000000 0.24083041
## [5,] 0.3942008 0.2463523 0.08986825 0.2408304 1.00000000
```

We can confirm that it is PSD by using this in our rmvnorm function:

```
rmvnorm(10, sigma=calc_cov_matrix(c(1,2)))[,1:5]
```

```
##              [,1]        [,2]        [,3]       [,4]        [,5]
##  [1,] -0.09314371 -0.43678450 -0.57082822 -0.3371184  0.08086846
##  [2,] -0.39140400 -0.92777883 -1.35414078 -0.9945356 -0.19110168
##  [3,] -0.56276741 -0.45540330 -0.89182799 -0.5807598 -1.76115053
##  [4,] -1.10008427 -0.26328500  0.51200875 -0.0941359  0.26478923
##  [5,]  0.27494848 -0.08861078  0.44855347 -0.4120130 -0.19458806
##  [6,]  1.89550902  3.14217078  2.32027870  2.8716105 -0.05089975
##  [7,] -0.43837280  1.33579099  1.65898275  1.7199681 -0.73991886
##  [8,]  2.26917297  2.28088632  1.52064692  1.8065199 -0.57068566
##  [9,] -0.27557378 -0.57744501 -0.08147708 -0.7706386 -0.38549811
## [10,]  0.10876317  0.35343622  0.63297681  0.4215467  0.77546086
```

It works!

Now, how to choose c,l, μ, σ^2? Let's maximize log likelihood! *Do not expect an output out of the code chunk below as we are just creating the function, not executing it yet*

```
nll = function(params){
  n = nrow(geo.dist)
  cov_matrix <- calc_cov_matrix(params[1:2])
  log_lik = mvtnorm::dmvnorm(x = Maine_temp$TMAX, mean = rep(params[3], n),
                             sigma = cov_matrix + params[4]*diag(n), log = TRUE)

  return(-log_lik)
}
```

Let's check to make sure our function works: *Here we are executing the function we created above*

```
nll(c(1,2,30,5))
```

```
## [1] 170.8368
```

We can now use the optim function:

```
optim(c(1,2,30,5), nll, lower=c(0.001, 0.001, 0,0),method='L-BFGS-B')
```

```
## $par
## [1] 67.961906  5.404865 34.941108  8.172643
##
## $value
## [1] 150.2727
##
## $counts
## function gradient
##       28       28
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Our estimate of μ here is similar to our empirical average. which is a good sanity check.

```
mean(Maine_temp$TMAX)
```

```
## [1] 31.17241
```

```
var(Maine_temp$TMAX)
```

```
## [1] 16.91712
```

We see that the empirical variance is higher than our estimated variance for the noise. This might indicate that some of the empirical variance is explained away by spatial covariances.

Let's store these optimal parameters: *Do not expect an output out of the code chunks below as we are just creating the parameters, not using them yet*

```
params_mle = optim(c(1,2,30,5), nll, lower=c(0.001, 0.001, 0,0),method='L-BFGS-B')$par
```

```
sigma_hat = params_mle[4]
mu_hat = params_mle[3]
cov_hat = calc_cov_matrix(params_mle[1:2]) + sigma_hat*diag(nrow(geo.dist))
```

Kriging / making predictions

Often times in spatial settings your goal is interpolation: based on data you've seen, make predictions at some unseen locations.

A statistical way to think about this is making a prediction at one location, conditional on all the other observations. In the multivariate normal setup we gave this problem, we can leverage a formula for the conditional mean in MVN: https://en.wikipedia.org/wiki/Multivariate_normal_distribution#Conditional_distributions (https://en.wikipedia.org/wiki/Multivariate_normal_distribution#Conditional_distributions)

Let's create a function to do this. We will use our existing covariance matrix, but just pretending like we were "missing" the temperature at one location: *Do not expect an output out of the code chunk below as we are just creating the function, not executing it yet*

```
calc_cond_exp = function(x, m, cov_matrix, idx) {

  mu_1 = m
  mu_2 = rep(m, length(x)-1)

  sigma_22 = cov_matrix[-idx,-idx]
  sigma_12 = cov_matrix[idx, -idx]

  a = x[-idx]

  # Return the conditional mean
  return(mu_1 + sigma_12 %*% chol2inv(chol(sigma_22)) %*% (a - mu_2))
}
```

As a way of testing the model, we can see how well we predict each location only using all the other locations: *Here we are executing the function we created above*

```
imputed_values = rep(NA, nrow(geo.dist))

for (idx in 1:nrow(geo.dist)) {
  imputed_values[idx] = calc_cond_exp(Maine_temp$TMAX, mu_hat, cov_hat, idx)
}
imputed_values
```

```
##  [1] 27.34007 30.63332 33.80307 31.37992 27.01284 28.81246 28.62361 28.98989
##  [9] 32.31363 34.35490 33.04728 28.98775 29.73557 33.96867 28.91399 29.96133
## [17] 28.08902 34.95951 33.54211 34.03521 32.55009 34.16492 26.45752 30.37681
## [25] 32.34918 31.04576 27.39515 33.95942 31.79514 31.54391 26.15299 31.81839
## [33] 26.66370 32.98767 30.15231 35.47576 27.75329 26.52449 30.41623 36.52912
## [41] 35.91860 32.64502 27.46943 30.96402 26.75274 32.13893 28.05625 36.06819
## [49] 31.84385 31.20401 31.99918 34.06808 34.80739 33.01435 30.94257 31.50045
## [57] 33.75630 29.54070
```

MSE of these imputed values:

```
mse = function(true, pred){mean((true-pred)^2)}

mse(Maine_temp$TMAX, imputed_values)
```

```
## [1] 8.647655
```

```
mse(Maine_temp$TMAX, mean(Maine_temp$TMAX))
```

```
## [1] 16.62545
```

The spatially aware model performs much better, with half the MSE!

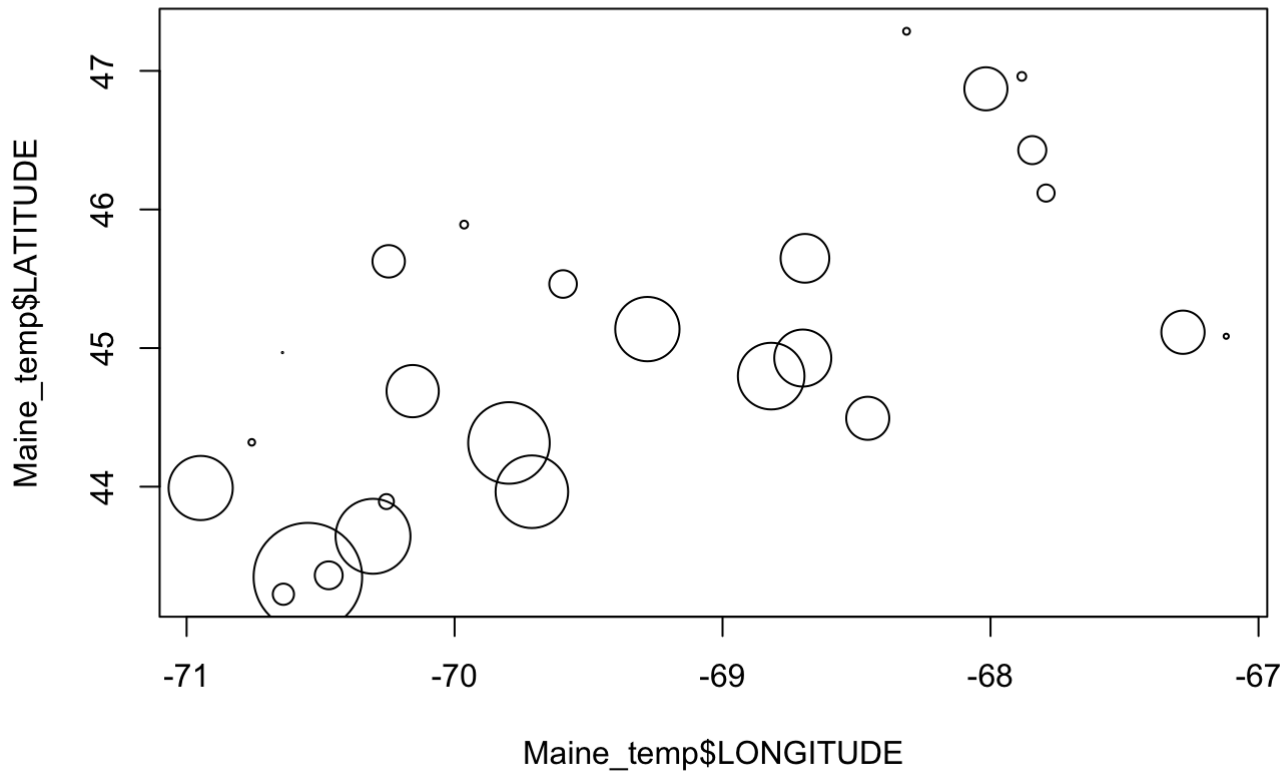
Using nearest neighbor interpolation performs even worse:

```
nearest_neighbor_idx = apply(geo.dist, 1, function(x){which(x==sort(x)[2])})

mse(Maine_temp$TMAX, Maine_temp$TMAX[nearest_neighbor_idx])
```

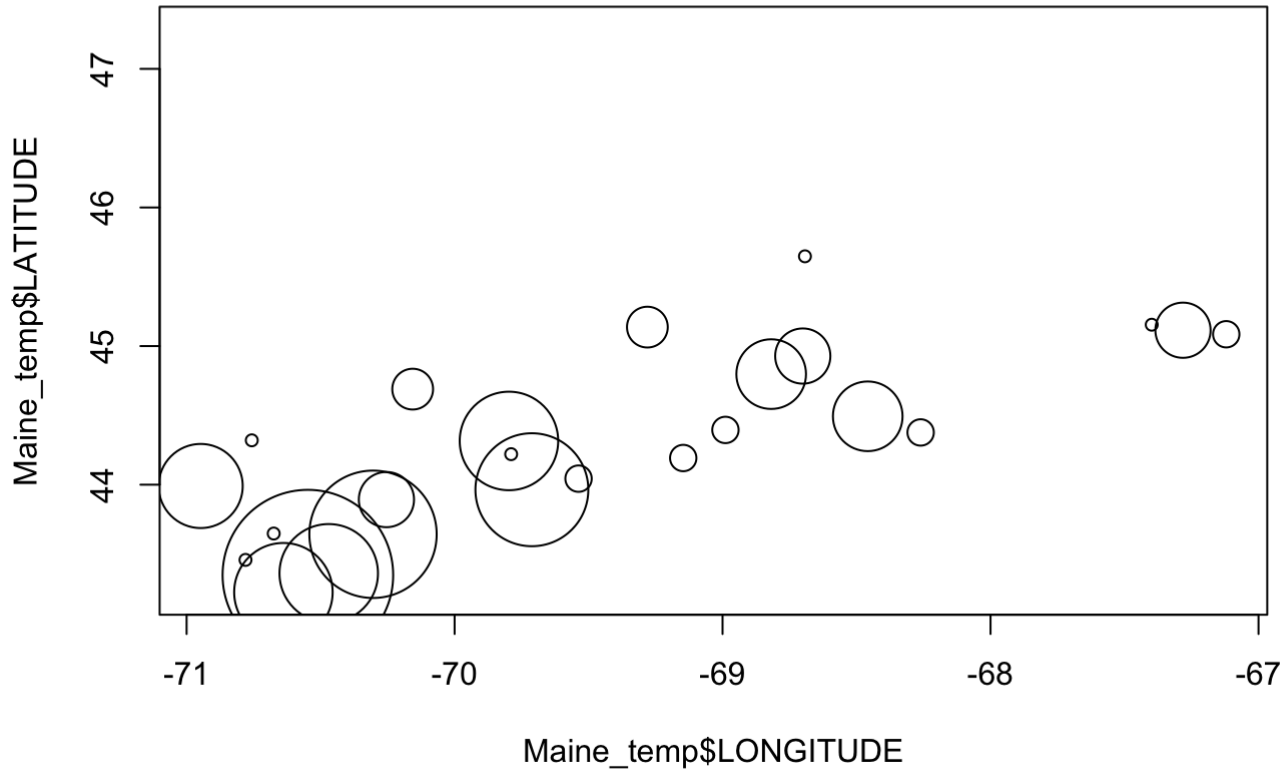
```
## [1] 19.5
```

```
plot(Maine_temp$LONGITUDE, Maine_temp$LATITUDE, cex = (Maine_temp$TMAX - imputed_values)
)
```



Are errors now independent of location?

```
plot(Maine_temp$LONGITUDE, Maine_temp$LATITUDE, cex = (Maine_temp$TMAX - mean(Maine_temp
$TMAX)) )
```




```
for (i in 1:20) {
  nearest_neighbor_idx = apply(geo.dist, 1, function(x){which(x==sort(x)[i])})

  print(mse(Maine_temp$TMAX, Maine_temp$TMAX[nearest_neighbor_idx]))
}
```

```
## [1] 0
## [1] 19.5
## [1] 12.74138
## [1] 21.7931
## [1] 22.10345
## [1] 21.67241
## [1] 21.41379
## [1] 19.18966
## [1] 26.58621
## [1] 22.67241
## [1] 20.31034
## [1] 18.58621
## [1] 15.24138
## [1] 29.68966
## [1] 27.44828
## [1] 19.22414
## [1] 23.05172
## [1] 25.87931
## [1] 20.86207
## [1] 34.2931
```

Pros and Cons of Gaussian Processes

Pros:

- Handles spatial covariance very naturally
- A statistical model, so you can create prediction intervals
- Can also handle spatiotemporal data (can make each covariance entry a product of a space kernel and a time kernel)
- μ doesn't have to be a simple linear

Cons:

- Very computationally expensive to fit. Time cost increases dramatically as you add more locations (the dmvnorm function has to invert the covariance matrix)
- Expensive to predict, since it is a nonparametric model. To make a prediction for a new point, we need to use all of the observations. In a parametric model (e.g. linear regression), once you know the parameters (e.g. coefficients) then you don't need the observations to make a prediction. Here, we need to invert a large covariance matrix to make a prediction.